Dissemination-Oriented Communication Systems: Final Report

Prof. David R. Cheriton
PRINCIPAL INVESTIGATOR
cheriton@dsg.Stanford.EDU
(415)-723-1131

November 26, 1996

1 Introduction

This is the final report for ARPA contract no. DABT63-91-K-0001. This contract was focused on exploring dissemination-oriented communication systems, systems in which the primary mode of communication was based on publishing information sources multicasting to subscribing sources. The approach was develop new protocols at the network, transport, session and presentation layers, and evaluate these protocol designs by implementation and use with a few sample dissemination applications that are also being developed. The objective was to design a next generation communication protocol architecture for dissemination-oriented communication that was applicable in a wide variety of (inter)networks, ranging from high-speed, high delay terrestrial to low-speed, high error rate mobile networks.

We started the contract with some basic ideas of how to revise the various protocol levels to support the dissemination model, based on our prior experience with IP multicast, VMTP and application-layer protocols. Over time, these ideas were refined such that some were incorporated into a common protocol, others were discarded as unworkable or suboptimal, and some were realized as originally proposed.

It would probably be politic to claim we are highly successful in every aspect of the original proposed work. However, usually some of the greatest insights and progress come from mistakes, and in that vein, we had lots of insights and progress. In this report, we also try to be honest and clear about how some of our original proposed directions failed, and why.

The following sections describe the results for each of the original deliverables, plus sections corresponding to what we actually developed, if that was different from originally proposed. We first describe some background ideas and how their development affected our work on this contract.

2 Additional Background

Several ideas from previous contract work represented starting points for our thinking or gave us a starting base of code, etc.

2.1 VMTP

The Versatile Message Transaction Protocol (VMTP) [25] was developed under a previous contract to explore the feasibility of a transport protocol that supported a range of multicast behaviors ranging from best-effort datagrams similar to UDP to reliable multicast to all receivers. Unlike just using UDP, VMTP provided all these behaviors within one protocol interface and one protocol implementation.

The greatest strain on the VMTP design occurred in trying to incorporate support for call streaming into the design. VMTP was centered on a request-response or transactional model of communication which was then extended at considerable complexity to support streaming. Over the course of the earlier work on the contract, the request-response basis also seemed less appropriate for dissemination-oriented communication behavior, which is typically streamed, multicast and unidirectional. Thus, the VMTP approach was phased out in favor of a more streaming-based approach to communication services.

2.2 Sirpent

The Sirpent (Source Internetwork Routing Protocol with Extended Network Transfer) was an attempt to revisit and extend source routing for high-speed switching through internetworks. It was inspired in part by earlier work we did on all-optical switches where state was at a premium, so passing the state (in the form of source routes) in each packet was attractive. We continued to experiment with some of the Sirpent ideas but the approach was superseded by the object-oriented RPC system we developed under this contract.

2.3 Memory-Based Messaging

Memory-based Messaging as a concept was originally conceived and refined under an earlier (D)ARP contract (Workform Processing under Tice Deyoung) which was focused on high-speed parallel processing. The basic idea with memory-based messaging is to treat a shared memory segment between two or more address spaces as a communication channel. A sending process writes a message into this shared memory region and the receiver(s) read the message from this shared memory region. The earlier contract implemented optimizations for memory-based messaging in an experimental multiprocessor system that we developed, called ParaDiGM. These optimizations included a memory-generated interrupt that informed the receiver(s) of a message being written, a cache consistency model oriented to messaging that reduced the consistency overhead of messaging in the memory system, and automatic signal-on-write, which generated the receiver signal automatically. The memory-based messaging mechanism was implemented and evaluated in the ParaDiGM architecture as part of the thesis work of Robert Kutter.

A key issue we struggled with at the beginning of the contract was resolving the "conflict" between the memory-based messaging model of communication and the source-routed datagram approach of Sirpent. Memory-based messaging is fundamentally a connection-oriented style of communication because it relies on a "connection" setup in the form of creating a shared memory segment between the sender(s) and the receiver(s). On the other hand, the source-routed datagram approach of Sirpent entails carrying state in the packet, namely the route, rather than expecting the forwarding switches to contain this state. In this sense, it is the opposite extreme to connection-oriented communication.

After significant experimentation and thought, we concluded that the connection-oriented model of memory-based messaging could be integrated into the datagram model, leading to the IP+ approach described in Section 6.1. This reduced the emphasis in the project on source routing, although source routing still has considerable appeal for technologies in which the state is expensive, e.g. all-optical switching nodes.

A significant step in this work was the extension of memory-based messaging to network interfacing, developing the idea of network virtual address space. This mechanism provides the basis for the VASSA networking interface, described in Section 5. It also led to the development of a new operating system kernel, the V++ Cache Kernel, using interprocess communication based on memory-based messaging.

The connection-oriented model prompted by memory-based messaging and its refinement and extension in VASSA lead to the IP+, a connection-oriented extension of IP. These accomplishments are discussed in the following sections.

3 Dissemination-Oriented Communication Model

The dissemination model of communication was developed considerably during the course of the contract, both in concept and in realization.

In the dissemination model, information sources uses channels to disseminate information to a potentially large and changing set of channel subscribers, analogous to, for example, cable TV. This model contrasts with the conventional two-party bi-directional conversational model and the more recent client-server request-response models. The work was motivated by the convictions that:

1. this form of communication will dominate in the future with uses in distributed interactive simulation, multimedia conferencing, and so on,
2. this form of communication is poorly supported by current protocols, and
3. a significantly improved architecture can and should be designed based on this model that also accommodates the other two models as special cases.

This model of taking single-source multicast as the base case for communication was the key inspiration for the contract and was a theme through out almost all of the work performed under the contract, as described below.

4 Sirpent Implementation

At the (inter)network layer, the focus was on an extended source-routing based protocol provides a high-degree of source control over communication services through control of the route and priority. It also integrates datagram network routing and virtual circuit routing into a common framework, as different points on the client/network state spectrum. Briefly, a hop in a source routing can be a physical hop or a logical hop. A logical hop can be a multi-physical hop path provided by network routing. Logical hops in an (inter)network correspond to virtual circuits, at least from the standpoint of addressing. Work is underway to investigate simple means for clients to respond to congestion and failures along network paths, with significant disruption and with minimal switch support. There is also work on incorporating rate-based congestion control into this scheme.

At the (inter)network layer, a routing server and network topology management library was been implemented that provides multiple routes to a specified destination based on network topology and link costs provided to the library. This facility is designed to support the extended source-routing approach we are exploring with the Sirpent/Viper protocols.

The Unix kernel implementation of the Sirpent protocol was extended to support encapsulation and transport of IP. An modified IP implementation has been extended to request routing information from the above-mentioned routing server, allowing traffic to selected IP addresses to be forced to go over Sirpent channels.

The work on Sirpent was de-emphasized as we recognized the difficulties in implementing source routing in switches, especially for multicast, and as the IP+ approach was recognized as being far more compatible with currently deployed Internet technology. However, the Sirpent work did foster the development of some of the key ideas behind IP+ and provide us with a far greater understanding of some of the issues in efficient protocol implementation.

5 High-speed Network Interfacing: VASSA

The memory-based messaging concept was extended under this contract to supporting memory-based messaging between network nodes using a novel network interface design we referred to as VASSA, Virtual Address Space Switching Architecture. The basic idea is to have the network interface implement what we refer to as a network virtual address space, and then have it map between network virtual address space and physical memory, just as a processor MMU maps between application virtual address space and physical memory. In essence, a message written by the processor to a memory-based messaging channel appears in the corresponding portion of physical memory, which is then transferred out over a fiber optic link with an address corresponding to its mapping into network virtual address space. On reception, the VASSA interface maps the address of an incoming packet from network virtual address space into the local physical address space. Applications with that physical memory mapped into their address space can then read the message. Moreover, the ParaDiGM mechanisms for signaling on write can be used to notify applications of the reception.

With this design, the VASSA network interface is basically two MMUs, mapping from the physical memory to network address space and vice versa. Thus, the hardware is fast, simple and similar to that used in processors.

The PI wrote a draft report on VASSA in 1992 [16]. Its publication has been held up while Stanford applied for a patent on this technology. At the time of writing, we have received first office action on VASSA and are hopeful of getting a strong patent granted. The approach bears some similarity to that of the Princeton Shrimp approach, which was developed concurrently. However, Shrimp does not provide mapping on reception, thereby limiting the memory protection provided by this approach between network nodes.

As work on this contract, we designed, implemented and tested a prototype VASSA interface for the ParaDiGM multiprocessor board we had developed earlier. This approach allowed us to capitalized on the memory-based messaging support on the ParaDiGM hardware as well as being able to couple the VASSA interface into the second-level cache bus, providing very low latency to the processor and memory.

The first prototype VASSA interface ran at 266 Mb/second and used standard Fiber Channel link-level components. We produced 8 of these prototype boards and used them to support development of the higher-level protocols for VASSA, including the object-oriented RPC described in Section 12.

A second generation of the interface was designed and implemented supporting a 1 Gb FCS VASSA interface, but this time connecting to the industry standard PCI bus.

One key innovation in this second generation interface was support for caching connection state. In particular, when a packet is received for a network virtual address that is not contained in the current mapping state in the interface, the interface generates an interrupt, causing the CPU to load such a mapping so that packet can be delivered, or else drop the packet. This extension nicely deals with the problem of having more network virtual address space than the physical device actually handles. Moreover, the solution parallels that used with virtual memory. The interrupt is similar to a page fault being signaled by a processor.

One aspect of the first version of the VASSA interface that was basically abandoned in the second generation was the use of fixed-length cells that corresponded in size to the cache line size of the processor. This approach was designed to minimize the complexity of handling data transfers by matching them to the unit preferred by the memory system. However, maintaining an exact match was difficult or perhaps impossible because of differences between machines, especially over time. Another problem was the significant overhead per packet of header and trailer bits at these small packet sizes. Our solution in this second generation of the VASSA interface was to switch to variable-length packets. Then, the interface has the extra logic for converting a packet into memory block transfer units on reception and vice versa on transmission. Happily, this step in conjunction with the move to PCI bus makes it less problematic to use the VASSA interface for standard commercial hardware systems.

Overall, the VASSA effort has produced two versions of the hardware and has catalyzed some significant inventions. With the advent of 1 Gb/second Ethernet and other fast network technologies, the VASSA design should be able to make some significant contributions to allowing packets to be transmitted and received at a host or endstation with minimal hardware demands. We are currently exploring ideas on ways to fit standard protocols and protocol headers into this network virtual address space and mapping mechanism.

6 Channel Adaptation Layer

The Channel Adaptation Layer (CAL) was proposed as an alternative to UDP for structuring protocols on. Part of CAL was to support delivery to a process rather than just to the host, like with IP. A second idea was providing a sequence number as part of the packet format, rather than having to add it, as with UDP.

The CAL development was significantly influenced by our work on memory-based messaging. The CAL layer was originally conceived of as a separate protocol layer, much like an extended UDP, implemented in the operating systems kernel. After some initial efforts in this original approach, we primarily collapsed this direction into the memory-based messaging class library that we developed to support VASSA and OO RPC. In particular, the memory-based messaging of VASSA incorporated the intra-host addressing and the sequence number into the network virtual address of the packet. That is, this address is sufficient to deliver the data directly into the right buffer location in the right address space(s) of the right applications. The sequence number was effectively extended by an evolving base checksum, as described in Section 12, so that the sequence number wrap was largely insulated from the size of the buffer being used for communication.

We also proposed to investigate channels as truly distributed objects. With our development of multicast objects in the OO RPC model (see Section 12), channels can be represented in this form, resulting in a nicely recursive design in which application-level multicast objects are built on these channel multicast objects.

The work on SS, SM, MM (single source, single destination), (single source, multiple destinations) and (multiple sources, multiple destinations) was mapped in the memory-based messaging model onto different types of memory-based messaging channels, plus extended with the notions of dynamically created versus static well-known channels. For instance, static node-to-node unicast channels can be viewed as the equivalent of IP source/destination pairs. Memory-based messaging can use "well-known" messaging segments for this type of communication, well-known at least in the sense of their addresses and access within the local operating system node.

This work was extended further to a proposed extension of IP to support connection-oriented communications, as described in Section 6.1.

6.1 IP+

IP+ is an extension to IP that smoothly incorporates connection-oriented multicast support into the standard IP multicast solution. As part of this contract work, we designed and implemented this protocol in a prototype implementation for Unix. Various performance measurements of this implementation were also performed.

Jonathan Stone, one of the Ph.D. students, is making IP+ the primary focus of his research program and has been responsible for much of the IP+ implementation. This protocol seems relatively straight forward to implement as an extension of the IP module, similar to the extension required for IP multicast.

IP+ cleanly integrates both datagram and virtual circuit communication in the same basic protocol hierarchy. Datagrams can be used for relatively lightweigth communication such as accessing a name server or reading the time. Connections are used to establish high-performance transfer paths between source and destination when the traffic warrants. For instance, a tail circuit router may limit the datagram traffic to some small percentage of the link bandwidth, keeping the rest of the traffic for connections that have been "accepted" by the receiving station.

The use of connection-oriented techniques at the (inter)network layer raises the issue of connection management. This work is described in the next section.

As an additional activity at the internetwork layer, support work continued during the contract in conjunction with Steve Deering at Xerox PARC, formerly a student in this group, to support the porting and updating of the IP multicast code. There has also been work supporting the so-called "mbone", the multicast backbone being used to provide an experimental base for a number of multicast experiments, including video and audio multicast of the IETF meetings. The porting work with IP multicast as well as VMTP is important to keep the technology available as OS versions and hardware advances.

7 Fault-tolerant Connection Management

We investigated IP+ with a strong awareness of the common problem with connection-oriented communication, namely the loss of state in an intermediary can cause the associated channel to fail, requiring the sender to reconstruct it "by hand". One of our key objectives was to develop a connection management protocol that is robust in the presence of failures. The other concern was the extra roundtrips and associated delay that another connection-oriented layer can add, potentially an extra delay per hop. Another objective that arose in the latter part of our effort was to use the OO RPC system we developed for the call setup mechanism. We believed that it was feasible to use asynchronous OO RPC's to avoid the per-hop extra roundtrip times and the call bundle mechanism to avoid the extra roundtrip between the network layer and the transport layer.

A fault-tolerant connection protocol was designed and implemented that exhibited these properties (although it did not use the call bundle mechanism). Unidirectional channel setup was handled as recursively nested asynchronous calls, one per hop, progressing to the desired endpoint. Bi-directional setup is provided as an optimization, creating the second channel as part of the return processing.

Each node maintained sufficient information about the next 2 nodes in each direction such that a single node/hop failure can be recovered from by cooperation between the two adjacent nodes. Basically, the adjacent nodes provide the requisite information to the failed node on its recovery so as to heal the connection. An extension would be to heal the connection by providing the state to an alternative route, an extension we planned but have not completed to date.

Recovery from a more complex failure is handled by the endpoints recreating the channel, using the same information used to create the channel in the first place. Higher-level protocols are designed to handle this situation, which is largely equivalent to a X.25 connection restart.

In our experience, this approach provided roughly the same reliability and performance as a datagram design because the most common failure, the single hop or link, was recovered from normally local to the problem, similar to the rerouting that occurs with a datagram network such as IP.

We went through several versions of this connection management protocol, with the final version being adapted to support IP+ as well as the VASSA high-speed networking.

A final issue that we investigated is the naming of channels and the relation of this naming to fault-tolerance. One would like the principal that if it is possible to setup a channel from node to node B, then the channel can be established from its name. This implies the naming of channels has to be handled by the switching nodes that are necessarily on this path. Otherwise, it is possible for the connection setup to be possible but for the naming mechanism to fail because of the node handling the naming being dead. IP has this property in essence because each router effectively implements the "name space" of the subnet connected to it. Therefore, a subnet is accessible by IP "name" (address) if and only if there is a path to that subnet. By designating new channels in IP+ by a (sourceIP, destIP) address tuple, we can achieve the same property, building on the IP routing mechanism. We did not get to the point of implementing this name support for VASSA.

8 Congestion Control

We came into this contract with a view that congestion control and flow control were a challenge with dissemination-oriented communication, and had proposed a number of ideas to explore. Unlike unicast communication, it does not make sense with multicast to slow down the source just because one receiver is slow. In the course of the contract, we explored a number of ideas in this area.

8.1 Feedforward Control

We explored the idea of feedforward congestion control in which packet loss is signaled to the receivers, who then can determine what actions to take, including high-level actions such as changing or reducing the number of channels being subscribed to.

After several attempts in this direction, the best approach proved to be just including a sequence number in all packets, as designed into the CAL level, and relying on receivers to detect congestion by noticing gaps in the sequence numbers indicating packet loss.

This approach was used successfully in the log-based receiver-reliable multicast (LBRM) protocol [21] described in Section 13.4. Here, a receiver can call back to a log server to get packets that it failed to receive due to congestion or other failures. The source can monitor the log server callback level to determine when there is significant traffic congestion, and possibly scale back its transmission rate accordingly.

8.2 Reservations

After some consideration, we concluded that resource reservations do not make sense for multicast, and perhaps for networking in general. Traffic is too dynamic in nature, especially with multicast, where the whole delivery tree can change significantly over time. Therefore, it is difficult, if not impossible, to reserve the bandwidth needed for the multicast tree in advance. Moreover, the network layer handling congestion has no basis to determine who deserves a reservation in preference over who and does not operate on the time scales desired by humans. For example, if the vice presidents of some large company need bandwidth for a video conference, they typically schedule the meeting weeks or months in advance. Thus, the network should accept advance reservations weeks or months in advance of the actual need as well. However, there is basically no infrastructure for dealing with actions in a network that are that far in advance or persist for that length of time. Moreover, most current reservation schemes do not even deal with the length of time that a reservation is needed for, yet advance reservations require these limits. Finally, networks do not currently incorporate notions of users, accounts and such like, so a whole additional authentication infrastructure is needed to tie reservation requests to user accounts and to tie packets to these reservations.

8.3 Rate-based Congestion Management

We investigated the use of rate-based congestion management techniques to provide stability at higher traffic loads, and validate these techniques in large scale using network simulation. One Ph.D. student, Michael Greenwald, has studied various congestion control techniques and performed simulations of rate-based congestion control techniques and related this work to work on window-based techniques. This work has formed the basis for our support for congestion management and accounting in VASSA [17].

With the VASSA interfaces, we incorporated hardware counters for packets and bytes, and used a periodic process to read and clear these counters, thereby providing a sum of traffic for accounting plus approximate rate information for congestion control. The rate information can be used to push back to the source in theory. However, in practice, it seems inappropriate to push back to a multicast source unless all packets are being dropped, for otherwise there are links that can handle the data rate. Consequently, we conclude that congestion control is best signaled again to the receivers by packet drop, and that hop-by-hop network layer flow control is best employed to just impose earlier drop on packets and flows that are hitting a downstream bottleneck. That is, it is better to drop the packet on entry to the network than have it dropped after several hops take it to the congestion point.

8.4 Early Drop

In this early drop model of hop-by-hop flow control, multicast often reduces the frequency of drop because normally at least one downstream port is able to keep up with the packet rate on the channel. One can regard this hop-by-hop flow control as effectively a variation on the broadcast-and-prune mechanism, where the pruning occurs because of excessive packet rate (or lack of downstream bandwidth), not because of lack of reception interest. In any case, the key observation is that multicast flow control seems best applied as a means of dropping packets earlier in the network, not a way of slowing down the source.

Building on this early drop model, the IP+ mechanism provides connection state in the network that can be back-traced to effect the early drop at the upstream node. Without this state, it is more difficult to determine where a given packet flow is coming from.

Applying this same thinking to a conventional reliable unicast protocol such as TCP, we see that TCP also assumes that congestion is signaled to the receiver by packet drop. The receiver than effectively communicates with the sender (by failing to acknowledge the dropped packets). The sender then chooses an action to take, which in the case of TCP, is to slow down. However, this approach is consistent with our model in that the sender can reasonably choose to slow down when reliable delivery is required and there is a single recipient.

Overall, multicast significantly changes the perspective on congestion control and flow control relative to many schemes proposed for unicast. Our "signal-the-receiver-with-drop" approach, relying on the receiver to take action and optionally the sender, makes the flow control truly an end-to-end issue. Then, network-layer flow control and congestion control is largely an optimization to cause packets to be dropped earlier on their path than otherwise, when they are destined to be dropped at some later bottleneck switch.

9 Dissemination Transport Protocol

At the transport layer, We proposed to develop a dissemination transport protocol (DTP), building on our experience with VMTP, and adapting these ideas to dissemination requirements.

In the early stages of the contract, a full-time systems programmer and some masters-level students have worked to generate two updated releases of VMTP, with extensions for multicast streaming in particular as well as numerous bug fixes.

In a separate effort, the PI and a Ph.D. student have generated a draft report [10] that described a basic design for a next generation of transport protocol builds on the lessons learned from VMTP and the major focus of this work on efficient dissemination support. (This protocol incorporated aspects of what was called CAL and DTP in the original proposal.) A first-level prototype of this design was implemented by the student over the summer of 1993, running over Ethernet as well as Fore's ATM networking technology.

This initial design work lead to several key ideas. First, we realized the merits of the recursive structuring of the protocol, using RPC calls for acks and other connection management purposes. Second, the work on memory-based messaging suggested a way to incorporate the sequencing into the addressing of data for packets, causing this functionality to be supported by a transport class library, not the kernel. Finally, we recognized the benefit of being able to specialize the transport protocol behavior for a variety of performance and functionality reasons, particularly for multicast.

The work on object-oriented RPC stub compilers in conjunction with our work on memory-based messaging indicated that transport protocol functionality could be incorporated into the OO RPC stub compiler and run-time support, rather than be implemented separate in the operating system, as is done conventionally. Consequently, these ideas and techniques were incorporated into our work on object-oriented RPC, as described in a following section. That is, the transport protocol was merged with the RPC protocol for performance, simplicity and ease of implementation.

We regard this change in approach as a significant innovation in itself. Rather than relying a separate, fixed and non-extensible kernel implementation of the transport protocol, and rather than having each application implement its own transport protocol, our revised approach provides a (class) library-based approach that provides a common implementation across applications yet is specialized by the stub compiler options and application-specific extensions to meet multicast and performance extensions required by particular applications.

10 Virtual Instruction Stream Protocol

At the presentation layer, we proposed to investigate the use of a virtual instruction stream protocol which extends the conventional notion of data description used for presentation protocols to full programming capability, similar to the programmability arising using Postscript as an interface language to window systems, such as NeWS. Developing this extra functionality was motivated by several factors. First, dissemination-oriented communication is often unidirectional, so sending responses back to solicit control actions is not feasible. Second, with multicast, callbacks can be expensive and subject to implosion-induced losses at the multicast source. Finally, cutting roundtrip times out by transferring control information has significant benefit in wide-area communications. The basic focus is on an object-oriented extensibility, building on the basic implicit model of presentation protocol, as represented by Courier and XDR, with the ability to define classes as units of procedural scope, and procedures within these classes.

In our work, we designed and implemented a basic virtual instruction set presentation protocol and a compiler that generates this presentation protocol for transmission over channels. Several experiments indicated that this VISP approach did significantly reduce the latency for performing some operations over the network by eliminating some round-trip times.

Building on this experience, we then redesigned the protocol to provide higher performance and to integrate it into a new object-oriented remote procedure call system that we developed, as described in Section 12. The virtual instruction stream aspect of this work was captured in the *call bundle* mechanism, that allowed encapsulation of calls as parameters for evaluation to other calls.

11 Memory-based Messaging OS: The Cache Kernel

Our work on extending memory-based messaging for network communication together with the high-speed demands of the VASSA network interface prompted a designing a whole new operating system kernel, the V++ Cache Kernel [3], succeeding we had developed under previous ARPA work, namely the V-system Kernel. The memory-based messaging allowed an implementation of interprocess communication in the kernel that was almost entirely realized as part of the virtual memory system. A significant portion of the complexity of a conventional virtual memory system could also be avoided using the application-controlled physical memory approach [18] we also developed during this contract. This integration into the virtual memory system also lead to the novel idea of having the kernel be a cache for operating system objects, rather than a complete implementation, the approach taken in previous so-called micro-kernel efforts.

The result of these ideas was the development of a new operating system kernel, the V++ Cache Kernel [3]. This kernel was specifically designed to exploit the memory-based messaging and VASSA networking facilities, providing a small PROM-able real-time kernel that supports multiprocessing and distributed systems. This operating system kernel supported all the VASSA-related software during the last two years of the contract. It is recognized as embodying some key new ideas in operating systems and is seeing some industrial interest in its deployment. We believe that our development of communication techniques that can substantially reduce the complexity of the most critical operating system component will be a substantial contribution to the robustness and security of future operating systems. especially for mission-critical systems in the military and civilian applications.

12 Object-oriented RPC

An object-oriented remote procedure call (OORPC) facility was developed that served to focus and integrate our work on transport, session and presentation protocols in support of dissemination, largely replacing the more independent approaches of CAL, DTP and VISP. This work was also integrated into our effort to develop highly efficient protocols and protocol implementations for the memory-based messaging facility provided by IP+ and VASSA.

This work produced a OORPC stub compiler, a "channel" library that implemented the transport layer support, an object directory mechanism, and an RPC class library that interfaces between the automatically generated stubs and the channel library. These are all working, demonstrable software components that are now being used in at least one commercial software development project.

Several novel aspects arose as part of this effort.

12.1 Integrated Layer Memory-Based Messaging

The OORPC stubs incorporated so-called integrated layer processing as part of the memory-based messaging interface. Consequently, on transmission, calls are marshalled, checksummed and copied to a communication channel all as part of one data move operation. Similarly, a call is demarshalled, checksummed and copied to a call stack from the communication channel in one data move operation. This approach minimized the amount of copying, a performance-limiting overhead at the network rates of interest.

12.2 Memory-based Sequencing

Memory addresses in conjunction with a "evolving base" checksum was used in place of convention packet sequent numbers. This approach, fitting with the memory-based messaging approach, eliminated the need to add and remove packet header information, further contributing to performance. The evolving base meant that a channel could use an address range that corresponded more to the buffer size than the sequence number wrap behavior. The epoch numbers encoded in the checksum base detected sequence number wrap in place of using long sequence number ranges.

The division between core and extended operations is made possible in part because we have made assumptions about the underlying model of communication, and optimized the core operations to perform well according to that model. The model that made sense to us was a streaming window, partly because it corresponded so well with the memory-based messaging angle. Regardless of the model, it is necessary to make such an operational split between core and extended functionality, and must be sufficiently general in order to avoid limiting flexibility.

12.3 Extensible Operations with In-line Core Interface

The channel interface used by the stubs was divided into core operations and extensible operations, the former being in-line functions that very efficiently map to channel functionality but do not support extensibility, and the latter being virtual functions that can be overridden for performance optimizations and functionality extensions.

The extensibility aspect was used to implement a variety of types of multicast and unicast behavior. This extensibility allowed us to support dissemination-oriented communication within the OORPC framework, integrating this form of communication with conventional RPC. For example, this extensibility was exploited to implement a log-based receiver-reliable multicast channel (see Section 13.4) to support multicast OORPC calls suitable for DIS-like applications. It was also used with significant performance specializations to implement highly efficient communication in in a version of a particle-based wind tunnel simulation, one of our test applications.

12.4 Multicast Objects

A fourth innovation was the definition "multicast objects", logical objects that are not implemented on a single host, but represent an interface between callers and callees, allowing multiple in the latter category. In brief, any number of "implementors" can register to implement a given multicast object interface. Each such registered entity receives the subsequent OORPC calls to this instance of the interface. This extension was a key step in supporting the dissemination-oriented communication we targeted as part of this work.

Multicast objects, which often represent dissemination-oriented components of a system, are novel in the fact that there is no single concrete instance of the object that acts as a server in the traditional client/server relationship. Rather, each recipient invokes its own implementation of the logical multicast object to specialize processing in response to disseminated events. This revised concept of ownership requires new models for object location and binding including new support from object directories.

The new directory structure is actually important for both of the above points. It is necessary to support the "distributed implementation" nature of multicast objects, and it is also necessary to ensure that specialized versions of objects/channels are bound correctly on each communication endpoint.

12.5 Distributed Shared Memory Pointers

A fifth innovation was the invention of distributed shared memory (DSM) pointers as part of the RPC system. This approach allows one to pass a pointer to an object in distributed shared memory without marshalling the object across the communication channel. Instead, the RPC run-time can check that the receiver(s) have access to the same DSM region, and pass the object by reference, simply marshalling the pointer rather than the object.

This approach was chosen after several significant false starts. As one, we first attempted to pass large data parameters using the control/data separation used with VMTP, but applied to memory-based messaging and OORPC. This approach ended up being complicated and rather non-intuitive in its performance implications, as parameters varied between fixing in the control portion and not. We were forced to reassess the approach when a performance-critical application we were implementing, a particule-in-cell wind tunnel simulation [24], used slightly larger data than fit in the control portion, leading to suboptimal performance.

The next approach we tried was more conventional, marshalling all of the data into the communication channel. Unfortunately, the marshalling cost ended up being significant in this approach, especially for simple data like file pages, etc. This cost was particularly pronounced with "local" RPC calls passing references to data that was in locally shared memory.

With DSM pointers, the RPC marshaller converts the local pointer to a global form, basically DSM segment name and offset, and the demarshaller converts it back. A stub routine that handles a DSM pointer includes a check with the system run-time that the pointer is shared with the receiver(s) of the call. Not all aspects of the new approach have been implemented at the time of writing, but the basic design has been worked out. Nevertheless, we feel we have explored a number of approaches and finally uncovered a novel one that seems better than previous solutions.

This is also another form of specialization — the ability to specialze the marshalling and demarshalling of particular parameters, on a per-parameter basis. Also another example of connections set-up in a sense, because each side needs to agree on the shared region before hand.

12.6 Call Bundles

A sixth innovation is the *call bundle*. A call bundle is a call that is encapsulated to be passed as a parameter to an OO RPC call, such that it is invoked at the calling site as a nested call within the OORPC call, providing a result to this call. The original motivation for call bundles was to eliminate multiple round-trip times in a layered architecture. In particular, a file open operation over a connection-oriented network can be bundled and passed as a parameter to the connect RPC, allowing the connection to be established and file to be opened in one roundtrip time. In this case, the bundled file open operation RPC is invoked in the context of the connect call after the connect processing has been completed.

The call bundle mechanism can be used to implement the full generality envisioned with the VISP approach. Call bundles can be nested so that a parameter to a bundled call can be itself a call bundle. A server can provide remote procedures that implement the basic control constructs of a programming language, such as if...then...else, while loop, block, for loop, etc. Procedures and variables can effectively be defined by remote procedures that return procedure and variable objects, which can in turn be nested in other call bundles.

This approach opens up the avenue of being able to bundle an entire program and pass it as a parameter to a remote procedure call, such as "execute", which would cleanly give you the entire functionality envisioned for VISP. However, this extension seems to incur a non-trivial cost over the basic call bundle mechanism. Unfortunately, we did not manage to fully address this issue of whether the extra generality was worth the cost, and also precisely what the cost is. It is simpler to just provide a basic call bundle facility without these control flow extensions, etc.

Call bundles also lead to larger data size for calls (multiple RPC headers) and therefore are another argument for integrated control and data and some of the other techniques used for large calls.

There were also some other uses of call bundles that were influenced in part by the object-oriented framework and the DIS objects in particular. First, there was often a disconnect between the granularity of the interface attributes and the granularity of network updates for efficiency. That is, an object in a DIS simulation might have position attributes and velocity attributes. These are separate from an interface standpoint, but you might want to batch updates to each of these attributes in a call bundle to avoid the overhead of multiple RPCs. This is essentially a separation between interface and wire format. Second, state changes often require rapid changes to particular attributes, which can be consilidated into a single, batched update rather than individual RPCs. Finally, many accesses to objects include a traversal of containing objects - plane to engine to propeller - that would require a round-trip and object import for each traversal on a traditional OORPC system. Call bundles allow the traversal to take place on the server, eliminating the round-trip and object import overheads of the traversal.

12.7 On-going Work

Our work in this area is continuing, albeit without dedicated funding at this stage. In particular, One of the Ph.D. students, Matt Zelesko, is working to complete his Ph.D. on this object-oriented RPC system. The final result should go well beyond our original goals of this contract, producing an object-oriented RPC system that incorporates support for multicast as well as more general specialization support for performance and functionality.

Overall, with OORPC, we completed the design and development of the object-oriented RPC and transport optimized for the memory-based messaging facility provided by IP+ and VASSA, thereby covering the original contract notions of a Channel Adaptation Layer (CAL), the Dissemination-oriented Transport Protocol (DTP) and the Virtual Instruction Stream Protocol (VISP), although in a somewhat different form than originally conceived. This work also covers what we proposed to complete in our no-cost extension statement of work.

13 High-level Dissemination Protocols

A key deliverable in the contract was prototype applications in one or more application domains that exercised the dissemination-oriented techniques developed in this contract. Over the course of the contract, our application direction became focused on distributed interactive simulation (DIS), both because of the rich set of challenges that this domain presents as well as the strong interest in military uses of this technology. There was also a strong interest from the students in this area.

13.1 ParaDISE

To this end, we developed the ParaDISE (Parallel Distributed Interactive Simulation Environment) software system, that implements an extensible distributed interactive simulation system on a collection of PCs or workstations. This system implements the protocols and techniques we have developed, allowing us to visually demonstrate their characteristics as well as providing a strong experimental evaluation.

13.2 Dead Reckoning

One key aspect of this work was the development of the position-history-based dead-reckoning protocol [19], which demonstrated how to perform efficient remote rendering of active simulation objects with reduction in bandwidth and error over previous approaches. Our work also provided a far more comprehensive study and evaluation of protocols in this area and the performance of our protocol than had been done previously.

13.3 Projection Aggregations

This work was extended further with the investigation of *projection aggregations*[20] as an approach to handling objects are various levels of detail in a distributed interactive simulation. Previous work had handled objects like a tank as a single entity, with articulated parts such as the turrret and firable objects such as missiles handled as special cases. Our work provides a unified approach that combines the best of levels of detail based on the organizational structure of entities, whether battalions of tanks or individual tanks for example, as well as the virtual geographical proximity of the objects.

One of the Ph.D. students, Sandeep Singhal, has made this work the focus of his Ph.D. and extended the techniques and evaluation considerably in the process. He has also served on a number of architectural review boards run by IDA for ARPA reviewing STOW and next-generation DIS directions.

13.4 LBRM

In support of DIS applications and others, we developed *log-based receiver-reliable multicast (LBRM)*[21]. This high-level protocol takes a novel approach to achieving reliable multicast delivery, placing the onus on the receiver to ensure delivery and relying on storage at a log server for recovery.

This protocol was designed, implemented and evaluated in a number of settings, with particular attention to its applicability for supporting dynamic terrain in DIS. It specifically addresses the problem of entities such as a bridge that may not change or move for long periods of time and suddenly be (for example) destroyed. LBRM is designed to efficiently deal with objects that change very infrequently but must notify a potentially large set of interested parties quickly when they do change.

Besides the basic protocol, we developed and implemented several extensions and refinements, such as variable heartbeat, distributed logging and statistical ack'ing, to allow the protocol to scale better.

One of the Ph.D. students, Hugh Holbrook, has made this work the focus of his Ph.D. and extended the techniques and evaluation in a number of important ways. In particular, he has looked at using LBRM for Web caching consistency as well as file caching. He has also served on a number of architectural review boards run by SRI for the army reviewing multicast technology and directions.

13.5 Area-of Interest Multicast

A limited amount of work was done designing and implementing a dynamic directory service that would support area-of-interest multicast for DIS. The basic idea of area-of-interest multicast is to subscribe to the multicast channels of entities that are located within your visual range, so you receive update packets from these entities, and not from those outside your area of interest. A active directory function is required to provide updates to entities as new entities and thus, multicast channels, enter and leave their area of interest.

This work originally focused on using relatively ad hoc multicast techniques. However, it in combination with the dynamic terrain problem lead our thinking towards LBRM. It nows seem preferable to implement this active directory service using LBRM, using a receiver-reliable multicast channel to deliver updates on entities/channels in the corresponding area of interest.

13.6 CATOCS

Another issue we investigated was the provision of more comprehensive ordering mechanisms in multicast protocols. Conventionally, protocols such as UDP and IP multicast provide no ordering information on the packet at all. Our transport approach was to provide ordering information per channel, with single-source channels. Others have proposed multicast protocols that support total ordering or causal ordering of messages originating from several different sources, examples include ISIS, Transis and TOTAL.

Our investigation [4], done with a colleague Dale Skeen at a local distributed systems company, concludes that causal and total ordering mechanisms in the communication layer are not effective in reducing mechanism, improving correctness or improving efficiency. After looking at all the types of multicast applications, we conclude that causally and totally ordered communication systems (CATOCS) do not appear like a sensible approach to distributed systems design. This negative "result" is important in raising some serious questions about the merits of further work on the CATOCS direction, which has been an active area of research for more than 10 years. Although we did not *prove* CATOCS was of no value, we clearly identified and illustrated a number of major limitations of this approach that were not widely recognized and should be given suitable attention before this approach is seriously considered for further funding or deployment.

Overall, we exercised the dissemination-oriented communication techniques developed under this contract in this application domain and extended and tested the dissemination-oriented protocol techniques we have developed for distributed interactive simulation (DIS), including the position-history-based dead-reckoning protocol and the area-of-interest multicast management protocol, as called for in our no-cost extension.

13.7 Directory/Routing Service

A prototype directory and routing service was developed to support source routing and Sirpent management, as proposed. However, the shift in focus away from source routing after the early stages of the contract reduced the benefits to pursue this aspect more fully.

14 Technology Transitions

Protocol designs and overall architectural ideas from this work were incorporated in the DARPA ISAT Summer Study on distributed interactive simulation as a result of the PI's participation.

Aspects of our work were presented at the DIS Standardization workshop held at Institute for Simulation and Training in Florida, September 1992.

The U.S. Army Research Institute, Department of the Army, has used Bolo, a distributed tank simulation program written by one of the students on the project using some dissemination ideas. The program is being used under the name of TWIST.

Protocol designs and overall architectural ideas plus the experience from this work were used as part of the PI's participation in two architectural reviews of work for STOW'94 and STOW'97 (in March'93 and Aug'93) for the ARPA AST office as well as in an Office of Technology Assessment Defense Modeling and Simulation Project, in which the PI is participating.

The V software that was initially developed under a previous contract and refined and extended further continues to be used outside of Stanford and distributed and licensed by the Stanford Office of Technology Licensing. This is a very effective way of reducing duplication of effort and improving transfer of results.

The VMTP protocol has been widely distributed, providing other ARPA contractors with a base for exploring distributed applications using a reliable and quasi-reliable multicast transport protocol. See (gregorio.stanford.EDU: ftp/vmtp-ip). VMTP was been proposed to SMPTE, a broadcast video production industry standards group, for use in coordinating multiple pieces of video equipment.

Stanford has invested in applying for patents on the VASSA and distributed shared caching inventions from this project, and is thus pushing to export this technology to commercial interests. There is no significant licensing to report at this stage because the patents have not been issued yet, but we are hopeful that patents will be granted and licenses will be forthcoming, particularly as network and processors speeds increase to make these inventions compelling.

The Cache Kernel microkernel is currently being used by one company as part of its embedded operating system, which has also resulted in the system being ported to a more popular architecture, namely the Power PC architecture.

New work is under way to port this system to the MIPS architecture. We are hopeful that this initial use will lead to other licensees.

The variable heartbeat mechanism has been of interest to those developing Internet mbone tools and we are told is being incorporated into the multicast session protocols in that arena. We are working to interest other groups such as the DIS community in the LBRM protocol itself. We think this work will be of significant interest as there is more use of dynamic terrain and large-scale deployment.

The object-oriented RPC mechanism with its specializations for performance and multicast is being used by one commercial enterprise at this time and there has been significant interest from others since a recent presentation of this work at the International Conference on Distributed Computer Systems (ICDCS) in Hong Kong in May 1996.

The Ethernet group membership protocol (EGMP) is being revised by the IEEE (renamed Group Address Registration Protocol) to adopt as an IEEE 802 protocol to allow scalable multicast at the MAC level.

The Virtual LAN Management Protocol (VLMP) which supports multiple multicast/broadcast domains distributed across a MAC-interconnect collection of LAN segments is being studied and revised for possible adoption by the IEEE in the virtual LAN area.

15 Funding, Equipment and Personnel

We summarize aspects of the funding, equipment and personnel over the life of the project.

15.1 Funding

Rather than rely exclusively on (D)ARPA resources for all our support, we actively sought and received additional sources of funding from industry that provided greater leverage on the funding. Several sources of funding have provided for more equipment and staff than provided for in the contract. IBM, Intel and Microsoft have donated equipment and software. We have also received significantly reduced prices on ECAD software from Cadence, Logic Automation and others. I believe that this funding has amplified the results we have achieved under the ARPA funding and provided funding for support personnel, allowing us to make our software more complete, more reliable and more efficient, i.e. faster. We also feel that it is speeding the transfer of the technology to industry where it can then become available in products to meet military requirements.

15.2 Equipment

Over the course of the contract, we have purchased and acquired through donations a variety of workstations, high-end PCs and server machines plus a switched hub and firewall. The equipment budget has been stretched as far as possible to using discounts, manufacturer equipment grants, etc. to provide a research environment of maximal effectiveness for the type of research we have been performing and plan to continue in the future.

15.3 Personnel

The contract has supported the PI, 15 Ph.D. students and 17 masters students in various degrees of support. Of these Ph.D. students, 10 have either completed their theses, or published significant research reports. We have been very successful in attracting good students as research assistants to the project. The number of students involved has been greater than the funding level would support because a number of the students we were initially supporting have now been awarded fellowship support (Hertz, IBM, NSF fellowships). Fortunately, these students have continued to contribute to the goals of the project even after being awarded fellowships.

The contract also supported a systems programmer for the work on VMTP, a woman who graduated as top-rated Masters student in our program, partially supported by the contract, who went on to develop a distributable version of VMTP. She has since gone on to a take significant role in protocol work in the industry. Similarly, we supported

3 hardware engineers over the course of the contract, 2 of which who have gone on to take positions in innovative companies developing high-speed networking products, exploiting the expertise they gained as part of this contract work.

In summary, a major contribution of the contract has been to transfer the innovative ideas and techniques of this work through a large number of young, highly capable students and engineers to industry.

15.4 Publications

The publications generated during the project are included here as the references for the above summaries of our work.

References

[1] David R. Cheriton, An Information Management Model for a University, working paper, Distributed Systems Group, Stanford University, 1995.

[2] David R. Cheriton, The ATM Phenomenon: Lessons to be Learned, working paper, Distributed Systems Group, Stanford University, 1995.

[3] David R. Cheriton and Ken Duda, A Caching Model of Operating System Kernel Functionality, 1st Usenix Symposium On Operating System Design and Implementation (OSDI), November 1994. p. 215-228.

[4] David R. Cheriton and D. Skeen, Understanding the Limitations of Causally and Totally Ordered Communication Systems, Symp on. Op. System Principles, Dec. 1993, Ashville, NC. pp 44-57.

[5] Michael Greenwald and David Cheriton The Synergy Between Non-blocking Synchronization and Good Operating System Structuring, 2nd Usenix Symposium On Operating System Design and Implementation (OSDI), October 1996. p. 215-228.

[6] D. R. Cheriton (With H.Goosen, Hugh Holbrook, and P. Machanick) Restructuring a Parallel Simulation to Improve Cache Behavior in a Shared-Memory Multiprocessor: The Value of Distributed Synchronization, *Proceedings PADS-93*, San Diego, California, May 1993.

[7] David R. Cheriton (with S. Singhal, M. Greenwald and J. Stone), Designing an Academic Firewall: Policy, Practice, Internet Society 1996 Symposium on Network and Distributed System Security, Jan. 1996.

[8] D.R. Cheriton, IP+: Integrating Connection-oriented Communication with IP, working paper, Distributed Systems Group, Stanford University, 1996.

[9] D.R. Cheriton, A Generalized Redirection Facility for IP based on Virtual Interfaces, working paper, Distributed Systems Group, Stanford University, 1996.

[10] D.R. Cheriton, A Multicast Transport Protocol for the next generation of Communication Systems, working paper, Distributed Systems Group, Stanford University, 1992.

[11] D.R. Cheriton, K. Duda, Logged Virtual Memory, 15th ACM Symposium on Operating Systems Principles, December 1995, Colorado, p. 26-39.

[12] D.R. Cheriton and R. Kutter, Optimized Memory-based Messaging, USENIX Computing Systems Journal, to appear. 1996 (also a Stanford Computer Science Technical Report).

[13] D.R. Cheriton and K. Harty, A Market Approach to Operating System Memory Allocation, MARKET-BASED CONTROL: A PARADIGM FOR DISTRIBUTED RESOURCE ALLOCATION, World Scientific, February, 1996.

[14] D.R. Cheriton, Network Technology Independent Encryption Device (N-TIED), Informal white paper to ARPA and NSA, Distributed Systems Group, Stanford University, 1993.

[15] D.R. Cheriton and M. Zelesko, Specializing Object-oriented RPC for High-Performance Applications, ICDCS, May 1996.

[16] D.R. Cheriton, VASSA: Virtual Address Space Switching Architecture, Distributed Systems Group, Stanford University, 1993. (also patent application filed through Stanford).

[17] D.R. Cheriton, Building Congestion Control on Accounting in a High-speed Packet Switched Network, Distributed Systems Group, Stanford University, 1993.

[18] D.R. Cheriton and K. Harty, Application-controlled Physical Memory using External Page-Cache management, ASPLOS, Boston, October 1992, pp 187-199.

[19] D.R. Cheriton and S. Singhal, Exploiting Position History for Efficient Remote Rendering in Networked Virtual Reality, *Presence: Teleoperators and Virtual Environments*, Vol 4, No 2, Spring 1995. Pages 169-193.

[20] D.R. Cheriton and S. Singhal, Using Projection Aggregations to Support Scalability in Distributed Simulation, Proc. 16th ICDCS, May 1996, p. 196-206. IEEE

[21] D.R. Cheriton (with H. Holbrook and Sandeep K. Singhal), Log-Based Receiver-Reliable Multicast for Distributed Interactive Simula tion," *Proceedings of SIGCOMM'95*, Pages 328-341. Published as *Compu ter Communications Review*, Vol 25, No 4, 10/95.

[22] D.R. Cheriton and C. Williamson, Loss-Load Curves: Support for Rate-based Congestion Control in High-Speed Datagram Networks, SIGCOMM'91, Zurich, September 1991, pp 17-30.

[23] D.R. Cheriton (with H.Goosen and A. Karlin), Chiron: A System for Parallel Program Performance Visualization, *Computer-Aided Design*, 1994.

[24] D.R. Cheriton, H.A. Goosen and P. Machanick, Restructuring a Parallel Simulation to Improve Shared Memory Multiprocessor Cache Behavior: A First Experience, Shared Memory Multiprocessor Symposium, April 1991, pp 23-31.

[25] D.R. Cheriton, VMTP: Versatile Message Transaction Protocol, RFC 1045, Feb. 1988 (developed originally under a previous contract.)

15.5 Other Activities

The PI served on the Internet task force on end-to-end services, chaired by Bob Braden, for most of the period of this contract.

The PI contributed to a number of (D)ARPA reviews of DIS technology through IDA. He also participated in an ISAT study.

The PI served as a consultant for a number of organizations interested in multicast technology as well as more general networking issues, including IDA, Teknekron Software Systems (now TIBCO), Apple, Sun, Granite Systems, IBM and Airsoft.

16 Follow-on Activities

A key point of evaluation of a significant research project is whether it leads to new and important directions of research. The dissemination-oriented communication effort has lead us to several new directions as well as others that are natural continuations of the work, as describe below.

There is on-going work on high-level protocols for distributed interactive simulation (DIS) with utility for the general area of distributed virtual reality systems. A number of students are working in this area and we hope to solicit additional funding to continue this work.

There is on-going work on network security firewalls with particular ideas on how to handle multicast. There is also further work underway on IP+. Finally, we see the potential to contribute to multicast routing protocols and the integration of switching and routing, addressing the performance challenges of supporting the Internet in the gigabit performance range.

17 Concluding Remarks

The Dissemination-Oriented Communication Systems project was a highly ambitious research effort exploring the fundamental shift of the computer communication paradigm from unicast to multicast. The consequence of this exciting venture was a large and diverse collection of research results, protocols and hardware/software realizations that have wide applicability to US military applications and to the commercial arena. We have summarized these results in this report and referenced the appropriate documents.

Of equal importance, we have produced a significant number of Ph.D. students in experimental Computer Science, addressing a pressing national need. In fact, lack of technical people in Computer Systems is recognized as a significant issue in addressing next generation military projects. For example, we proudly note that Steve Deering, the chief designer of IP version 6 and IP multicast, is a graduate of our research group. The quality of education of these students and the quality of the research in general would have been impossible without the funding support and careful research management we have enjoyed from ARPA on this project.

We have also invested more than typical in good software and hardware development, but produced more than normal return in the form of: (1) utility of the prototypes, (2) depth of experience with our ideas and their implementation, and, (3) transfer of technology to others. As a consequence, software implementing the protocols described in this report and our publication has been distributed to various universities, military sites and companies.

In addition to the immediate results of the contract work, we have created an research group and environment that is capable of (and of course willing to) take on advanced research in further areas of communication, parallelism and distributed computation of interest to ARPA and required for future military applications.

18 Acknowledgements

I wish to thank Ira Richter to taking the original initiative to fund this work, and to Mike St. Johns who took over as program manager and guided and supported us further to make a significant contribution to ARPA's goals in the areas of networking and distributed computing.